Computational Physics

Peter Uwer
AG Phänomenologie der
Elementarteilchenphysik (PEP)

Vorbemerkung

- Studienpunkte
- Merkblatt
- Form der abgebenen Lösungen
  - Gedruckt unter Verwendung der Latex-Vorlage
  - Kurz und knapp aber gleichzeitig klar und vollständig
  - Form ist relevant z.B. Achsenbeschriftung etc.
- Bei Unklarheiten (V/UE): bitte fragen !
- Falls Folien schlecht lesbar (Größe, Farbe etc, bitte mitteilen)
- Quiz

Vorbemerkung

Ergänzungsvorlesung

Effizientes Computing

Vorbesprechung:

Montag, 19.04.2010, 14:45 in NEW 15 3‘101

Warum Computational Physics ?

- Für viele Probleme ist keine analytische Lösung bekannt → numerische Methoden

  Bsp: Numerische Integration, Gittereichtheorie

- Ausdrücke in analytischen Rechnungen oft zu groß, um von Hand verarbeitet zu werden

  Bsp: Computeralgebra in Störungstheorie

- Analyse von umfangreichen Datensätzen (Experiment / Simulation)

  Bsp: Datenanalyse Large Hadron Collider (LHC), CERN → PetaByte (Mega,Giga,Tera,Peta)

- …

Was soll dieser Kurs leisten?

Computer als wichtiges und unerlässliches Werkzeug der modernen Physik einführen

Was soll dieser Kurs leisten ?

Zentraler Aspekt von „*Computing*“:

Algorithmen

Algorithmus ist verantwortlich für

- Funktion
- Numerische Stabilität der Lösung
- Laufzeitverhalten / benötigte Zeit

→ Es werden exemplarisch Algorithmen vorgestellt und untersucht, um in die für die Rechneranwendungen wichtigen Aspekte einzuführen

Was soll dieser Kurs leisten?

Schwerpunkt liegt auf numerischen Methoden,

d.h. insbesondere keine Computeralgebra

Numerik liefert (mehr oder weniger) Zahlen,

z.B. Wert eine Integrals

Woher weiß man, dass das Ergebniss richtig ist ?

„Weil man keinen Fehler gemacht hat“ ist keine gute Antwort!

Tests / Validierung ist extrem wichtig, gilt auch für algebraische Methoden / Computeralgebra

Gutes Verständnis des Algorithmus und der Anwendung / Physik kann helfen, sinnvolle Tests zu machen, im allg. schwierig

Was dieser Kurs nicht leisten kann/soll

- Kein Programmierkurs !
- Kein „Hackerkurs“

Arbeitsumgebung

CIP Pool

48 x Intel Core 2 Quad CPU 2.66GHz, 8GB Ram, 80 GB HD

Batch-System, Linux (Debian 4.0)

Laserdrucker

→ Performantes System das eine Vielzahl von Anwendungen möglich macht

Wie sage ich dem Computer was er machen soll ?

→ Programmiersprachen

„Programmiersprachen“

Sehr viele verschiedene Sprachen verfügbar !

Unterscheidungskriterien:

- Native/compilierte Sprachen ← → interpretierte Sprachen
- Interaktiv
- prozedural ←→ objektorientiert
- spezialisiert ← → universell
- …

Bsp.: Pascal, Cobol, Fortran, C/C++, Java

Macsyma, Maple, Mathematica, Form, Reduce,

Matlab, Scilab, Octave,

Phyton, Perl, Php, Csh, Bash

Welche „Sprache“ soll man verwenden ?

Im allgemeinen keine universelle Antwort möglich, abhängig von:

- Problemstellung
- Eigene Vorlieben / Fähigkeiten
- Aufwand (Einarbeitung, Umfang der Lösung, „Performance“ der Lösung)
- Verbreitungsgrad / Stabilität

Im Idealfall beherrscht man mehrere Sprachen und wählt jeweils das optimale Werkzeug

→ für Anfänger schwierig zu realisieren…

Im Kurs: Matrix Laboratory --- Matlab

- Leicht zu lernen, einfach erste Ergebnisse zu erzeugen
- Ermöglicht sich schnell auf die algorithmische Komponente bzw. Anwendung/Physik zu fokussieren
- Umfangreiches Hilfssystem
- Nützliche Fehlermeldung
- Wird auch von Prof. Wolff in CP II im Wintersemester 2010 verwendet werden

Matlab: Matrix Laboratory

Vorteile:

- Interaktiv → kurze Entwicklungszyklen
- Viele nützliche, gut getestete Algorithmen für unterschiedliche Problemstellungen
- In vielen Fällen intuitive zu benutzen, keine Kenntnis nativer Programmiersprachen nötig
- Erweiterbar
- Nützliche Zusatzqualifikation

Nachteile:

- Interaktiv
- In der Anschaffung teuer
- kein „home use“ für Studenten
- Syntax, kein „Type checking“
- Performance?